MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
755 Page Mill Road
Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street
Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900
Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway
Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. CV 10-03561 WHA

ORACLE'S OPENING CLAIM CONSTRUCTION BRIEF

Dept.: Courtroom 9, 19th Floor
Judge: Honorable William H. Alsup

Tutorial: April 6, 2011, 1:30 p.m.

Hearing: April 20, 2011, 1:30 p.m.

TABLE OF CONTENTS

Page

INTRODUCTION 1

LEGAL STANDARD 3

ARGUMENT 4

I. The Play Executing Step (’520 Patent) 4

II. Intermediate Form Code (’104 Patent)
Intermediate Form Object Code (’104 Patent) 7

III. Resolve / Resolving (’104 Patent) 9

IV. Symbolic Reference (’104 Patent) 12

V. Reduced Class File (’702 Patent) 14

VI. Computer-Readable Medium (’104 Patent)
Computer-Readable Storage Medium (’720 Patent)
Computer-Readable Medium (’520 Patent)
Computer Usable Medium (’702 Patent)
Computer-Readable Medium (’447 Patent)
Computer-Readable Medium (’476 Patent) 16

A. ’104 patent 18

B. ’720 patent 20

C. ’520 patent 21

D. ’702 patent 21

E. ’447 and ’476 patents 22

TABLE OF AUTHORITIES

Page(s)

CASES

Agilent Techs., Inc. v. Affymetrix, Inc.,
567 F.3d 1366 (Fed. Cir. 2009) 11

Astra Aktiebolag v. Andrx Pharms., Inc.,
222 F. Supp. 2d 423 (S.D.N.Y. 2002) 24

Ecolab, Inc. v. FMC Corp.,
569 F.3d 1335 (Fed. Cir. 2009) 25

Energizer Holdings, Inc. v. Int'l Trade Comm'n,
435 F.3d 1366 (Fed. Cir. 2006) 6

Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.,
535 U.S. 722 (2002) 25

In re Beauregard,
53 F.3d 1583 (Fed. Cir. 1995) 16

In re Lowry,
32 F.3d 1579 (Fed. Cir. 1994) 18

In re Nuijten,
500 F.3d 1346 (Fed. Cir. 2007) 18

Kathrein-Werke KG v. Radiacion y Microondas S.A.,
No. 07 C 2921, 2010 WL 2011939 (N.D. Ill. May 17, 2010) 5

Nystrom v. Trex Co.,
424 F.3d 1136 (Fed. Cir. 2005) 3

Phillips v. AWH Corp.,
415 F.3d 1303 (Fed. Cir. 2005) (*en banc*) ..... passim

Rhine v. Casio, Inc.,
183 F.3d 1342 (Fed Cir. 1999) 24

Trading Techs. Int'l, Inc. v. eSpeed, Inc.,
595 F.3d 1340 (Fed. Cir. 2010) 25

TABLE OF AUTHORITIES
(continued)

Page(s)

OTHER AUTHORITIES

EXAMINATION GUIDELINES FOR COMPUTER-RELATED INVENTIONS (1996) 23

THE IEEE STANDARD DICTIONARY OF ELECTRICAL AND ELECTRONICS TERMS (6th ed. 1996) 24

THE ILLUSTRATED DICTIONARY OF ELECTRONICS (7th ed. 1997) 24

MANUAL OF PATENT EXAMINING PROCEDURE § 2106.01 (8th ed. 6th rev. 2007) 16

MICROSOFT PRESS COMPUTER DICTIONARY (2d ed. 1994) 10, 13

MICROSOFT PRESS COMPUTER DICTIONARY (3d ed. 1997) 24

WEBSTER'S NEW WORLD DICTIONARY OF COMPUTER TERMS (5th ed. 1994) 13

INTRODUCTION

The technology covered by Oracle's seven patents relates to the Java Platform.[1] Java was originally developed by Sun Microsystems, which was acquired by Oracle in January 2010 and was renamed Oracle America, Inc. In the early to mid-1990s, Java rose to popularity with the rise of the Internet and the World-Wide Web. It was designed to achieve "**write once, run anywhere**" in a new, networked computing environment that included the Internet.

The basic idea behind Java is that a software developer writes application source code once, compiles it into an intermediate form known as Java "**bytecode**", and distributes the bytecode in the form of "**class files**" or "**jar files**." A user wanting to run the application can get a copy of the application bytecode through a variety of mechanisms, including by downloading it from the Internet. The user runs the code using a Java "**virtual machine**" that was written for the user's particular computer architecture. The same application code can be run on any computer that has a Java Virtual Machine. This is unlike other computer languages like C or C++, which are compiled to a particular processor's instruction set, and so must be separately compiled for each target platform. Consistency among each platform's Java virtual machine implementation is achieved because each conforms to the Java Virtual Machine Specification so it will run the Java bytecode as expected.

Because of its strengths, Java quickly became one of the most popular software platforms. Oracle estimates that the Java Platform has attracted more than 6.5 million software developers, and that more than 1.1 billion desktop computers and 3 billion mobile phones run Java. *See* http://www.java.com/en/about/ (last visited Mar. 17, 2011).

Oracle distributes the various components of the Java Platform, and permits others to do so, under a variety of licensing terms. There are commercial, royalty-bearing Java licenses, and

[1] We use "Java Platform" to distinguish from the Java *programming language*. Core pieces of the Java Platform include:

- The Java programming language
- The Java Runtime Environment (including Java Virtual Machine)
- Extensive Java Class Libraries and APIs (application programming interfaces)
- The Java Development Kit (collection of programming tools, including a Java compiler)

there are Java licenses that are royalty-free (e.g., OpenJDK code[2] is available under the GNU General Public License version 2). But not even the "free" versions of Java are licensed without important restrictions. The various licenses covering Java share one common goal: protecting compatibility. As the Java Community Process website explains: "A marketplace flooded with proprietary code that varies from the official specification, or worse, parallel platform versions, would 'fork' the platform, eliminating its basic 'Write Once, Run Anywhere' compatibility foundation." http://jcp.org/ja/press/news/licensing_update (last visited Mar. 17, 2011). Oracle, and Sun before it, has gone to great lengths to prevent forks—*Sun v. Microsoft* probably being the most famous example. The Java licensing model is an important part of that effort.

As Sun software engineers were developing the Java Platform, they faced many technical challenges. By today's standard, the desktops of the '90s were constrained computing environments, and in order to maximize performance, Sun engineers developed many innovative techniques to increase execution speed while using memory more efficiently. Furthermore, since Java was developed with the Internet in mind, security (e.g., protecting user data from being stolen and misused by malicious software) was a primary concern and an active area for Sun innovation.

Sun's Java Platform development efforts to date have resulted in about two thousand patents. The seven patents asserted in this case can be categorized as follows with respect to the computing environment constraints discussed above:

Improving execution speed: RE38,104; 6,910,205; 7,426,720

Efficient memory utilization: 5,966,702; 6,061,520; 7,426,720

Improved security mechanisms: 6,125,447; 6,192,476

The Court ordered the parties to select no more than six phrases to brief and argue at the claim construction hearing. Oracle selected two and Google selected four.

Oracle selected "the play executing step" from the '520 patent because it is potentially outcome-determinative for the claims in which it appears. Oracle contends that the meaning of

[2] *See* http://openjdk.java.net/faq/ and http://openjdk.java.net/legal/gplv2+ce.html.

the phrase is readily determined from the intrinsic record; Google contends the phrase is not amenable to construction and renders the claims indefinite because it lacks antecedent basis.

Oracle selected "intermediate form code"/"intermediate form object code" from the '104 patent because the construction of that term will significantly streamline the validity issues with respect to that patent. Google's overbroad construction is designed to capture the disparaged prior art disclosed in the specification as part of the background to the invention; Oracle's proposal will exclude most, if not all, of Google's asserted prior art references.

Google selected "symbolic (data/field) reference" and "resolve"/"resolving" from the '104 patent, and selected "reduced class file" from the '702 patent. Oracle believes that Google's proposed constructions for these terms are intended to support motions for summary judgment of non-infringement, although how precisely those motions might be framed we cannot say, as Google has not explained the grounds for any such motions in its interrogatory responses.

Google selected "computer-readable medium/"computer usable medium"/"computer-readable storage medium" for construction. These phrases appear in six of the seven patents. Google's proposed construction for these terms are intended to support a motion for summary judgment of invalidity under 35 U.S.C. § 101 for claiming nonstatutory subject matter.

LEGAL STANDARD

This Court is well aware of applicable claim construction law. Claim construction is a question of law for the Court. *Nystrom v. Trex Co.*, 424 F.3d 1136, 1141 (Fed. Cir. 2005). A claim term is to be given "the meaning it would have to a person of ordinary skill in the art at the time of the invention." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (*en banc*).[3] Disputed terms are to be interpreted in light of the claims themselves, the patent specification, and the prosecution history. *See id.* at 1312-14. While extrinsic evidence "may be useful to the court, . . . it is unlikely to result in a reliable interpretation of patent claim scope unless considered in the context of the intrinsic evidence." *Id.* at 1319.

[3] Unless otherwise indicated, all citations and internal quotations have been omitted, and all emphasis has been added. Patents are referred to by their last three numbers, and citations to patent specifications are in the form "patent, column:line", such as "'104, 2:60-62".

ARGUMENT

I. THE PLAY EXECUTING STEP ('520 PATENT)

Claim Phrase	Oracle's Proposed Construction	Google's Proposed Construction
the play executing step	"The play executing step" in claims 3 and 4 is a reference to the "simulating execution" step in claim 1	Indefinite – cannot be construed.

The phrase "the play executing step" appears in the preambles of asserted dependent claims 3 and 4 in the '520 patent. Each begins: "The method of claim 1 wherein the play executing step includes the steps of . . . ." Google argues that the claim is indefinite because "the play executing step" has no antecedent basis. But in fact it does—it is the "simulating execution" step:

> 1. A method in a data processing system for statically initializing an array, comprising the steps of:
>
> compiling source code containing the array with static values to generate a class file with a clinit method containing byte codes to statically initialize the array to the static values;
>
> receiving the class file into a preloader;
>
> **simulating execution** of the byte codes of the clinit method against a memory without executing the byte codes to identify the static initialization of the array by the preloader;
>
> storing into an output file an instruction requesting the static initialization of the array; and
>
> interpreting the instruction by a virtual machine to perform the static initialization of the array.
>
> 3. The method of claim 1 **wherein the play executing step includes the steps of**:
>
> allocating a stack;
>
> reading a byte code from the clinit method that manipulates the stack; and
>
> performing the stack manipulation on the allocated stack.

'520, 9:47-10:4. Claims 4 and 5 have the same preamble as claim 3.

The intrinsic evidence makes clear that the "play executing step" in claim 3 is a reference to the "simulating execution" step of claim 1. In the '520 patent, "simulating execution" and "play executing" are synonymous. Indeed, in the "Summary of the Invention" section, "play executes" is defined as "simulates executing": "the preloader identifies all <clinit> methods and simulates executing ('play executes') these methods to determine the static initialization performed by them." '520, 2:65-3:1. In the detailed description, the reverse can be found: "When play executing, discussed below, the preloader simulates execution of the byte codes contained in the <clinit> method by the virtual machine." '520, 4:64-66. The specification uses the phrases interchangeably to refer to the same act:

'520, 4:36-41	'520, 6:23-27
Processing consistent with the present invention is performed by the preloader 220 searching for a <clinit> method, and when it is found, the preloader (1) simulates execution of the <clinit> method to determine the effects it would have on memory if it was interpreted by the virtual machine 222	The preloader receives as a parameter a method information data structure that defines the <clinit> method, described in the Java™ Virtual Machine Specification at pp. 104-106, and play executes the byte codes of this <clinit> method.

Given this disclosure, there can be no doubt that "play executing" in claim 3 refers to "simulating execution" in claim 1. *See Kathrein-Werke KG v. Radiacion y Microondas S.A.*, No. 07 C 2921, 2010 WL 2011939, at *5 (N.D. Ill. May 17, 2010) (construing "stripline sections", "stripline segments", and "stripline elements" to be the same, because the intrinsic evidence showed patentee "obstinately" used the terms to mean the same thing).

The prosecution history supports this construction. The patent examiner also viewed "play execution" and "simulated execution" as being synonymous. In the examiner's summary of an October 13, 1999 interview, he wrote: "The general background of the invention was discussed. In particular, the disclosure of the 'pre-loader' and 'simulation' function as disclosed in claims 1, 6, and 12." '520 file history, 10/13/1999 Examiner Interview Summary (Declaration of Marc David Peters in Support of Oracle's Opening Claim Construction Brief ("Peters Decl.") Ex. 1). On that date, Claims 1, 6, and 12 all recited "play executing," which the examiner referred to as "simulation."

The reason that claim 1 says "simulates execution" and claim 3 says "play executing" is because of an amendment to claim 1 that the examiner agreed would "further clarify the distinctions between the claim and the cited art." '520 file history, 10/18/1999 Amendment at 2 (Peters Decl. Ex. 2). That amendment changed the claim as follows:

> simulating execution of [play executing] the byte codes of the clinit method against a memory without executing the byte codes to identify the static initialization of the array by the preloader;

Id. The point of the amendment was to include in claim 1 the concept of "without executing the code" that was present in other claims that had already been allowed by the examiner (e.g., claim 6: "play executing the code without running the code"; claim 18: "simulating execution of the code without running the code"). When the amendment to claim 1 was made, no corresponding amendment to claims 3, 4, or 5 was made, so they still recited "wherein the play executing step." But of course the understanding of which step in claim 1 was the "play executing" step did not change—it was the same step it was before.

The meaning of the claim is reasonably ascertainable, particularly given the way in which "play execution" and "simulated execution" were used interchangeably both in the specification and in the prosecution history. The use of the phrase "wherein the play executing step" does not render the claim insolubly ambiguous or incapable of construction. "When the meaning of the claim would reasonably be understood by persons of ordinary skill when read in light of the specification, the claim is not subject to invalidity upon departure from the protocol of 'antecedent basis.'" *Energizer Holdings, Inc. v. Int'l Trade Comm'n*, 435 F.3d 1366, 1370-71 (Fed. Cir. 2006) (holding that claim was not indefinite because "anode gel" provided the antecedent basis for "said zinc anode").

II. INTERMEDIATE FORM CODE ('104 PATENT) INTERMEDIATE FORM OBJECT CODE ('104 PATENT)

Claim Phrase	Oracle's Proposed Construction	Google's Proposed Construction
intermediate form code **intermediate form object code**	executable code that is generated by compiling source code and is independent of any computer instruction set	code that is generated by compiling source code and is independent of any computer instruction set

The phrase "intermediate form object code" appears in asserted claims 11, 12, 17, 22, and 23; and "intermediate form code" appears in asserted claims 19, 20, 21, 27, 28, 29, 30, 31, 32, 33, 34, 35, 39, 40, and 41. In the '104 patent, "intermediate form object code" and "intermediate form code" are synonymous; that is not in dispute. Nor do the parties dispute that "intermediate form code" is the output of a compiler or that it need not be tied to any particular computer architecture or instruction set. The fundamental dispute is whether each phrase refers to code that is executable. It does.

Plain and ordinary meaning controls this dispute. *See Phillips*, 415 F.3d at 1312-13. As recited in the claims, each phrase means executable code that is generated by compiling source code and is independent of any computer instruction set. For one example, claim 11 recites:

> 11. An apparatus comprising:
>
> a memory containing **intermediate form object code** ***constituted by a set of instructions***, certain of said instructions containing one or more symbolic references; and
>
> ***a processor configured to execute said instructions*** containing one or more symbolic references by determining a numerical reference corresponding to said symbolic reference, storing said numerical references, and obtaining data in accordance to said numerical references.

According to claim 11, "intermediate form object code" is a set of instructions that a processor is configured to execute. *Phillips*, 415 F.3d at 1314 ("To begin with, the context in which a term is used in the asserted claim can be highly instructive."). In other words, "intermediate form object code" is executable. Every other claim in which "intermediate form code" or "intermediate form object code" appears consistently refers to it as executable: claim 12 ("interpreting said

instructions in accordance with a program execution control"); claim 17 ("a method for executing said program"); claim 19 ("memory for use in executing a program"); claim 20 ("computer-implemented method for executing a compiled program containing instructions in an intermediate form code"); claim 21 ("such that when the program is executed"); claim 22 ("a processor configured to execute the instruction"); claim 23 ("a method for interpreting a compiled program in intermediate form object code"); and claims 27-35, 39, 40, and 41 ("execute the program" or "executing the program"). The qualifier of "intermediate form" applied to "code" and "object code" is to indicate that the code is not dependent on the instruction set of a specific type of machine, as is the typical case for compiled object code. *Compare* '104, 1:25-29 ("In a compiled programming language, a computer program (called a compiler) compiles the source program and generates executable code for a specific computer architecture.") *with* '104, 1:58-61 ("In an interpreted language, a computer program (called a translator) translates the source statements of a program into some intermediate form, typically independent of any computer instruction set.").

The specification supports the conclusion that intermediate form code is executable. The Summary of the Invention explains that the claimed invention covers a "method and apparatus for generating ***executable code*** and resolving data references in the generated code . . . ." '104, 2:25-29. The specification consistently reinforces that "intermediate form [object] code" is executable:

> Therefore, under the present invention, the "compiled" intermediate form object code of a program achieves ***execution performance*** substantially similar to that of the traditional compiled object code, and yet it has the flexibility of not having to be recompiled when the data objects it deals with are altered like that of the traditional translated code, since data reference resolution is performed at the first execution of a generated instruction comprising a symbolic reference.

'104, 5:41-49.

Google's proposed construction is inconsistent with the claims and the specification. Google seeks to embrace what the '104 specification calls an "intermediate representation," which is an internal state of a compiler rather than executable code output by a compiler. *See, e.g.*, '104, 4:12-32 & Figs. 4-5. The specification explains that "intermediate representation" and "intermediate form code" are not the same:

> Shown in FIG. 4 is one embodiment of the compiler 42 comprising a lexical analyzer and parser 44, an intermediate representation builder 46, a semantic

> analyzer 48, and a code generator 50. These elements are sequentially coupled to each other. Together, they transform program source code 52 into tokenized statements 54, intermediate representations 56, annotated intermediate representations 58, and ***ultimately intermediate form code 60*** with data references made on a symbolic basis.

'104, 4:15-23. Figures 4 and 5 show that source code transformed to an intermediate representation as part of the compilation process is later output as intermediate form object code after it goes through a code generator. The claims reinforce the conclusion that an "intermediate representation" is different from "intermediate form code". Claim 21 recites "intermediate form code containing symbolic field references associated with an intermediate representation of source code for the program, the intermediate representation having been generated by lexically analyzing the source code and parsing output of said lexical analysis." Unsurprisingly, this corresponds to the description of Figure 4 reproduced above.

By leaving out the requirement that intermediate form code be executable, Google's proposal is an effort to broaden the '104 claims to include within the scope of "intermediate form code" the intermediate representations employed by prior art compilers. This broadening is not supported by the specification, file history, or the claim language, and should be rejected.

III. RESOLVE / RESOLVING ('104 PATENT)

Claim Phrase	Oracle's Proposed Construction	Google's Proposed Construction
Resolve / Resolving	No construction necessary. "Resolving" a symbolic reference is determining its corresponding numerical reference.	replace/replacing at least for the life of the process

The term "resolve" or "resolving" appears in a majority of the asserted claims of the '104 patent. "Resolution" is the determination of the location that corresponds to a given name. The plain language of the claims provides the meaning of "resolving" such that no construction is necessary. Claim 20,[4] for example, states: "resolving the symbolic reference in the instruction by determining a numerical reference corresponding to the symbolic reference." The idea behind

[4] The text of claims 20, 21, and 23 appears in the April 29, 2003 certificate of correction to the '104 patent.

"resolution" is that, although computers store and access data by location (e.g., memory locations or addresses), people like to refer to data by name (e.g., x, y, name) when writing code, so it can be better understood. For a computer to run a program, it needs to "resolve" the names (the symbolic references) in the program to determine the corresponding locations (the numeric references). This process is reflected in a dictionary definition that Google offered in support of "symbolic reference": "Symbolic address: A memory address that can be referred to in a program by name rather than by number. The interpreter, compiler, or assembler translates the name into the number that specifies the address." MICROSOFT PRESS COMPUTER DICTIONARY 379 (2d ed. 1994) (Peters Decl. Ex. 3); *see also* Joint Claim Construction Statement ("JCCS") (Dkt. 91) at 9. The translation referred to in the definition is a form of resolution.[5]

If the Court does construe this term, the Court should adopt Oracle's proposed construction because it is compatible with all the specification's uses of the word in connection with different types of programming languages, and it provides meaning to all relevant claim terms. Google's proposed construction, by contrast, is contrary to both the claims and the specification.

The different ways that the specification uses the word "resolve" or "resolution" shows that a symbolic reference is resolved by determining a corresponding numerical reference. *Phillips*, 415 F.3d at 1313 (claim terms must be read "not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification."). According to the '104 specification, in some programming languages, a compiler resolves references when it generates executable code, and before it is executed:

> In a compiled programming language, a computer program (called a compiler) compiles the source program and generates executable code . . . . References to data in the generated code are ***resolved prior to execution*** based on the layout of the data objects that the program deals with, thereby, allowing the executable code to reference data by their locations.

'104, 1:25-32. Thus a compiler may resolve references even for code that never comes to "life" by being executed. So Google's proposed construction that "resolving" is for the life of a

[5] The converse is not true, however. Resolution does not require translation.

particular process is contradicted by the specification.

The specification also discloses that, in some interpreted programming languages, "[e]ach of the symbolic references is resolved during execution ***each time*** the instruction comprising the symbolic reference is interpreted." '104, 2:3-6. This means the resolved symbolic reference is not ***replaced***—otherwise it would not need to be resolved again. So Google's proposed construction that "resolving" means "replacing" is contradicted by the specification.

Oracle's proposed construction encompasses determining the numerical reference corresponding to a symbolic reference, regardless of when or how often resolution occurs. Thus it fits equally well with compiled languages, interpreted languages, and languages that share characteristics of both, as with the '104 invention. '104, 2:16-24.

In contrast, Google's proposed construction should be rejected because it would render claim language superfluous. *Agilent Techs., Inc. v. Affymetrix, Inc.*, 567 F.3d 1366, 1378 (Fed. Cir. 2009) ("A claim construction that gives meaning to all the terms of the claim is preferred over one that does not do so."). In several instances, the claims and the specification include "resolve" in close proximity to "replace" or to synonyms of replace, indicating that "resolve" and "replace" have separate meanings. For example, claim 30 recites: "*replacing* each instruction in the program with a symbolic data reference with a new instruction containing a numeric reference *resulting from* invocation of a . . . *routine to resolve* the symbolic data reference." Claims 31 and 32 follow an almost identical pattern. The Summary of the Invention, moreover, describes the acts of resolving and rewriting separately within the same sentence: "The . . . routine . . . *resolves* a symbolic reference and *rewrites* the symbolic reference into a numeric reference." '104, 2:44-47. In yet another example, the original claim 1 of the 1992 application—part of the patent specification—recites, within a single step (c), "*resolving* said symbolic references to corresponding numeric references, *replacing* said symbolic references with their corresponding numeric references, and continuing interpretation." U.S. Patent App. No. 07/994,655 Original Claims at 14 (Peters Decl. Ex. 4); *see also* '104, claim 1. Had the inventor intended resolving and replacing to be synonymous, roughly half of the language quoted above would be unnecessary.

This analysis is further supported by the doctrine of claim differentiation, in which "the presence of a dependent claim that adds a particular limitation gives rise to a presumption that the limitation in question is not present in the independent claim." *Phillips*, 415 F.3d at 1315. A construction that equated "resolving" and "replacing" would render claim 14 identical in scope to claim 13, from which it depends. Claim 13 includes the limitation "resolving a symbolic reference in an instruction, said step of *resolving* . . . including the substeps of: determining a numerical reference corresponding to said symbolic reference, and storing said numerical reference in a memory." Claim 14 adds the limitation: "wherein said substep of storing said numerical reference comprises the substep of *replacing* said symbolic reference with said numerical reference." If "resolving" meant "replacing," as Google proposes, claim 14 would be completely superfluous. Such a construction must be rejected. *Id.* at 1324-25.

IV. SYMBOLIC REFERENCE ('104 PATENT)

Claim Phrase	Oracle's Proposed Construction	Google's Proposed Construction
symbolic (data/field) reference(s)	No construction necessary. The ordinary meaning is "a reference by name"	a dynamic reference to data that is string- or character-based

Variants of the phrase "symbolic reference" appear in each asserted claim of the '104 patent. The plain and ordinary meaning of "symbolic reference," "symbolic data reference," and "symbolic field reference" is a reference to something by name, so it need not be construed. The '104 patent contrasts a situation in which "references to data are made on a symbolic basis," such as when an instruction "references the variable y by the symbolic name 'y'", with a situation in which those references are resolved and code can "reference data by their locations." '104, 1:31, 1:64-67. The '104 patent specification has three examples of symbolic references (x, y, and name ('104, 1:51-54 & Fig. 1B)), but these examples are not limiting. *Phillips*, 415 F.3d at 1323; '104, 5:34-36 ("***As illustrated***, a data referencing instruction, such as the LOAD instruction 14", is initially generated with a symbolic reference, ***e.g. "y"***."). So, references in the '104 patent are either by location (a numeric reference) or by name (a symbolic reference).

Extrinsic evidence confirms that a "symbolic reference" is a reference by name. According to Google's own proffered dictionaries, a "symbolic address" is a "memory address that can be referred to in a program by name rather than by number. The interpreter, compiler, or assembler translates the name into the number that specifies the address" and a "symbol table" is a "list of names used in a program with brief descriptions and storage addresses." MICROSOFT PRESS COMPUTER DICTIONARY 379 (2d ed. 1994) (symbolic address) (Peters Decl. Ex. 3); WEBSTER'S NEW WORLD DICTIONARY OF COMPUTER TERMS 561 (5th ed. 1994) (symbol table) (*id.*); JCCS at 9. No construction of this term is really necessary.

Google asks the Court to construe "symbolic reference" to mean something that is not supported by the intrinsic evidence. First, Google proposes to construe "symbolic reference" as one that is string- or character-based. The significant problem with Google's approach is that "string" does not appear in the claims, specification, prosecution history, or Google's proffered extrinsic evidence. Nor is there any mention of "character" in a way that is germane to "symbolic reference." There's no reason to import these limitations into the claim.

Second, Google also proposes that the Court construe "symbolic" as "dynamic." How will this help the jury? How shall they know what makes a reference "dynamic"? Accepting Google's proposal threatens to require the Court to "construe the construction" should the meaning of "dynamic" be disputed later. Google's proposal again imports a limitation from the specification into the claim. The specification teaches:

> As shown in FIG. 7, upon receiving a data reference byte code, block 86, the main interpretation routine determines if the data reference is static, i.e. numeric, or ***dynamic, i.e. symbolic***, block 88. If the data reference is a symbolic reference, branch 88b, the main interpretation routine invokes the dynamic field reference routine, block 90.

'104, 5:10-15. The specification defines "dynamic" as symbolic, but not the other way around. In other words, a "dynamic reference" is a "symbolic reference," but there is no clear indication in the specification that a "symbolic reference" has to be a "dynamic reference."

Surely Google's proposed construction is designed to set up a non-infringement argument of some kind or another. Because Google still has not identified or described its argument in response to Oracle's interrogatories, Oracle cannot discuss how it relates to the proposed

construction. Regardless, Google's construction is divorced from the '104 patent specification and should be rejected.

V. REDUCED CLASS FILE ('702 PATENT)

Claim Phrase	Oracle's Proposed Construction	Google's Proposed Construction
reduced class file	No construction necessary. A "reduced class file" contains a subset of the code and data contained in a class file	a class file containing a subset of the data and instructions contained in a corresponding original class file

The phrase "reduced class file" appears in every asserted claim of the '702 patent. So far as Oracle has been able to determine through the meet and confer process, the dispute between the parties is about whether the "reduced class file" is itself a "class file." The parties agree that a "reduced class file" contains a subset of the code and data contained in a corresponding class file. No construction is necessary because this idea is inherent in the word "reduced," a word that the jury can understand. The general idea of the invention is to eliminate code and data duplicated among class files, and place only one copy of duplicated code and data in a multi-class file along with the non-duplicated code and data. '702, Abstract, 5:6-17. However, the '702 patent is specific about what a "class file" is, and a reduced class file does not qualify. (The parties dispute the meaning of "class file" as well, and it's not clear to Oracle why Google selected one phrase for construction but not also the other, since the two phrases are so intimately related.)

The class file structure is outlined in Figure 3 and in column 7 of the '702 patent:

> Embodiments of the invention can be better understood with reference to aspects of the class file format. Description is provided below of the Java class file format. Also, enclosed as Section A of this specification are Chapter 4, "The class File Format," and Chapter 5, "Constant Pool Resolution," of *The Java Virtual Machine Specification*, by Tim Lindholm and Frank Yellin, published by Addison-Wesley in September 1996, ©Sun Microsystems, Inc.

'702, 7:21-28. When source code is compiled, the definition of each class is stored by the compiler in its own class file, even if more than one class was defined in one source code file. *See, e.g.*, '702, 3:29-32, 7:31-32 ("A single class or interface file structure is contained in the class file."). This is why the '702 patent mentions the disadvantages that result from "the fact that

a typical Java application can contain hundreds or thousands of small class files." '702, 4:2-4. As mentioned above, the complete specification of the Java class file format is part of the '702 specification, and may be found in columns 11 through 43.

"Reduced class files" do not satisfy the class file format specification. The key issue is that a "class file is self-contained," but examples of reduced class files described in the specification violate that restriction. '702, 4:3-4. As the '702 patent explains, in one embodiment, "duplicated constant elements are placed in the shared constant pool as a shared element, and an element of the new constant type replaces the duplicated element in the reduced pool to direct constant resolution to the shared element in the shared constant pool." '702, 9:58-62. The "new constant type" contains an "index into the shared constant table." '702, 9:56-58. So in this embodiment, the pre-processor places in the reduced class file an index that refers to a shared constant pool entry ***outside*** the reduced class file. '702, 9:63-65. But class files must be self-contained, and do not—and cannot—contain indices to information stored elsewhere. Indeed, this is one of the disadvantages of class files that the '702 patent solves: each class file has to contain its own constants, even though different class files may have the same constants, creating duplication and inefficiency. '702, 3:65-4:14; *see also* '702, 4:45-60 (although a Java archive (JAR) file encapsulates multiple class files, they "remain subject to storage inefficiencies due to duplicated information between files").

In contrast, reduced class files are not self-contained. Although of course reduced class files can be very similar to class files, and their information can be stored similarly, embodiments of the '702 patent involve non-standard extensions to the class file format that permit code and data from one class to index into a constant pool shared among classes. '702, 9:65-10:5.

In the final analysis, the description of the embodiments in the specification makes clear that a "reduced class file" has duplicated information removed ("duplicated constant elements are placed in the shared constant pool as a shared element") but also may have non-standard elements added, such as new constant types and tags. '702, 9:55-65. So if a construction of the term is necessary, it should be just that a "reduced class file" contains a subset of the code and data contained in a class file.

VI. COMPUTER-READABLE MEDIUM ('104 PATENT) COMPUTER-READABLE STORAGE MEDIUM ('720 PATENT) COMPUTER-READABLE MEDIUM ('520 PATENT) COMPUTER USABLE MEDIUM ('702 PATENT) COMPUTER-READABLE MEDIUM ('447 PATENT) COMPUTER-READABLE MEDIUM ('476 PATENT)

Claim Phrase	Oracle's Proposed Construction	Google's Proposed Construction
computer-readable medium / computer-readable storage medium / computer usable medium	a storage device for use by a computer	any medium that participates in providing instructions to a processor for execution, including but not limited to, optical or magnetic disks, dynamic memory, coaxial cables, copper wire, fiber optics, acoustic or light waves, radio-waves and infra-red data communications

Six of the seven asserted patents contain claims using the phrases "computer-readable medium," "computer-readable storage medium," or "computer usable medium." These claims are often referred to as "Beauregard claims," after *In re Beauregard*, 53 F.3d 1583 (Fed. Cir. 1995). This type of claim gained popularity after *Beauregard,* in which a patent applicant challenged a Board of Patent Appeals' rejection of software-related claims. Before the Federal Circuit could rule, the United States Patent and Trademark Office changed its position, stating that "computer programs embodied in a tangible medium, such as floppy diskettes, are patentable subject matter under 35 U.S.C. § 101 and must be examined under 35 U.S.C. §§ 102 and 103." *Id*. at 1584. Based on this changed position, the PTO moved to dismiss the appeal, and the Federal Circuit vacated the Board's decision and remanded. *Id*.

Following *Beauregard,* a software invention claimed as a program embodied in a tangible medium has been consistently recognized as statutory subject matter under § 101. *See* MANUAL OF PATENT EXAMINING PROCEDURE § 2106.01 (8th ed. 6th rev. 2007) (explaining that "a claimed computer-readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of

the computer which permit the computer program's functionality to be realized, and is thus statutory.") (Peters Decl. Ex. 5). Beauregard claims allow inventors to claim as patentable subject matter: computer programs embodied on a floppy disk, hard disk, CD, DVD, computer memory, and similar storage media. Beauregard claims can be economically significant, in that they can be directly infringed by software companies in ways that computer system claims, requiring physical hardware, may not be.

The issue presented to the Court is this: some of the asserted patents (the '104, '520, '720, and '702) describe only storage media as embodiments of computer-readable media, the others (the '447 and '476) describe both storage media and transmission media. Google's claim construction approach is to take a passage from the '447 and '476 patent specifications and apply it to ***all*** of the patents-in-suit. One mistaken result, as explained below, is that Google is using 1997 evidence to construe a 1992 patent. Oracle submits that the proper way to construe a patent claim is to apply the *Phillips* methodology to that claim, considering first ***that*** patent's intrinsic evidence (its claims, specification, and prosecution history), and considering second the evidence extrinsic to that patent (such as other patents and dictionaries), because the "best source for understanding a technical term is the specification from which it arose, informed, as needed, by the prosecution history" and external evidence, while useful, is "less significant than the intrinsic record in determining the legally operative meaning of claim language." *Phillips*, 415 F.3d at 1315, 1317.

Under the proper methodology, the construction of "computer-readable medium," "computer-readable storage medium," and "computer usable medium" in the '104, '520, '720, and '702 patents is straightforward. Each patent discloses only storage media as embodiments, so the correct construction is "a storage device for use by a computer."

The construction of "computer-readable medium" in the'447 and '476 patents is more complicated. Although these patents disclose both storage media and transmission media (such as coaxial cable and fiber optics), the Court should decline to construe "computer-readable medium" to cover transmission media and therefore avoid a potential invalidity issue.

In 2007, the Federal Circuit addressed whether a "signal" was patentable subject matter. *In re Nuijten*, 500 F.3d 1346, 1353 (Fed. Cir. 2007). *Nuijten* concerned an appeal from a decision of the Board of Patent Appeals and Interferences. The PTO had found that a claimed "storage medium having stored thereon a signal with embedded supplemental data" was a "manufacture" and patentable, but that a claim directed to the signal itself was not. *Id.* at 1351-53. The court agreed, holding that the "claims on appeal cover transitory electrical and electromagnetic signals propagating through some medium, such as wires, air, or a vacuum. Those types of signals are not encompassed by any of the four enumerated statutory categories: process, machine, manufacture, or composition of matter." *Id*. at 1352.

We expect Google to argue that the "computer-readable medium" claims are invalid under Section 101 for being drawn to ineligible subject matter. But there are no "signal" claims in this case, and there is no question that a storage medium such as a computer memory is an article of manufacture or a composition of matter, and therefore patentable subject matter. *See In re Lowry*, 32 F.3d 1579, 1582, 1584-85 (Fed. Cir. 1994). What the *Nuijten* court did not address is how a district court should construe a "computer-readable medium" claim in an issued patent, when the specification describes both storage and transmission media. As explained below, the correct approach is to construe the phrase to mean only storage media, and preserve its validity.

We now apply the *Phillips* claim construction methodology to each patent in turn.

A. '104 patent

Claims 12 and 23 of the '104 patent claim a "**computer-readable medium** containing instructions for controlling a data processing system to perform a method . . . ." The Court should construe a computer-readable medium in the '104 patent to mean "a storage device for use by a computer." This construction is supported by the claim language, the specification, and the prosecution history of the '104 patent.

When construing a patent claim term, it has "the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention, i.e., as of the effective filing date of the patent application." *Phillips*, 415 F.3d at 1313. The '104 patent claims priority to U.S. Patent Application No. 07/994,655, which was filed in 1992. As such, the term

"computer-readable medium" must be construed as if by a person of ordinary skill in the art in 1992. References from 1997 (and particularly references from patents dating to 1997, which may reflect patent lawyer drafting preferences rather than the meaning attached to a term by skilled artisans) are therefore irrelevant to the claim construction inquiry.

In the specification, all the examples of computer-readable media are storage media: memories or storage devices to store code for use by a computer. '104, 3:52-56, Fig. 2 ("As shown in FIG. 2, the exemplary computer system 20 comprises a central processing unit (CPU) 22, a memory 24, and an I/O module 26. Additionally, the exemplary computer system 20 also comprises a number of input/output devices 30 and a number of storage devices 28."). Nowhere does the specification suggest that these computer-readable media are "coaxial cables, copper wire, fiber optics, acoustic or light waves, radio-waves and infra-red data communications" or other transmission media as suggested by Google's proposed construction.

The prosecution history confirms that a "computer-readable medium" in the '104 patent is a storage device for use by a computer. A claim directed to computer-readable media was first introduced in the file history in 1996 in the '204 reissue application, the parent reissue application to the '104 reissue application. In listing his reasons for filing the reissue application, the applicant stated that a "further error was that the claims may be challenged by an infringer as not reading literally on computer program code devices embodying the invention. Applicant is informed and believes that subsequent to the Commissioner of Patents' change of position in *In re Beauregard*, claims more specifically directed to computer program code devices are now permissible. I believe I have the right to further specifically claim computer program code devices which embody my invention as supported by the original disclosure." '104 file history, 11/21/1996 Reissue Application Declaration of James Gosling at 3 (Peters Decl. Ex. 6).

The applicant particularly pointed out and distinctly claimed the subject matter of a Beauregard claim: a computer program embodied on a tangible medium. His stated intent was to capture the statutory, patentable subject matter identified by a Beauregard claim. Oracle's proposed construction—a storage device for use by a computer—is supported by the intrinsic evidence. Google's proposed construction is not.

Google's proposed construction for the '104 patent does not make sense in light of the claim language, specification, or prosecution history. None of the intrinsic evidence or contemporaneous extrinsic evidence identifies "coaxial cables, copper wire, fiber optics, acoustic or light waves, radio-waves and infra-red data communications" as '104 patent embodiments. Therefore, it would be improper to construe the claims to cover such things.

B. '720 patent

Claim 19 of the '720 patent claims a "**computer-readable storage medium** holding code for performing the method according to claim 10." The Court should construe computer-readable storage medium in the '720 patent to mean "a storage device for use by a computer." This construction is supported by the plain meaning of the claim language and the specification.

The invention of claim 19 is a storage medium that holds code. Memories and disks are storage media that hold code. "Coaxial cables, copper wire, fiber optics, acoustic or light waves, radio-waves and infra-red data communications" are not storage media, and they do not hold code. In the specification, all the examples of computer-readable storage media holding code are memories or storage devices for use by a computer:

> Each client 13 is operatively coupled to a storage device 15 and maintains a set of classes 16 and class libraries 17 . . . .
>
> a server 18 is operatively coupled to a storage device 19 in which globally shareable class libraries 20 are maintained.
>
> Upon initialization, the master JVM process 33 reads an executable process image from the storage device 35 . . . .

'720, 4:32-34, 4:40-42, 5:48-49. Figures 1 and 2 illustrate the storage devices holding code. The specification does disclose computing devices connected by hardwired and wireless networks, but does not describe the networks as being storage media holding code. Indeed, transmission media cannot be storage media. Storage media store. Transmission media transmit. Transmission media do not hold code. Google's proposed construction, which includes transmission media, is contrary to the claim language. It would be improper to import such limitations as "coaxial cables, copper wire, fiber optics, acoustic or light waves, radio-waves and infra-red data

communications" into the construction of computer-readable storage medium for the '720 patent. The '720 claim language and specification only support "a storage device for use by a computer."

C. '520 patent

Claim 18 and its dependent claims 19-23 of the '520 patent claim a "**computer-readable medium** containing instructions for controlling a data processing system to perform a method . . . ." Like computer-readable storage medium in the '720 patent, this phrase means "a storage device for use by a computer." This construction is supported by the plain meaning of the claim language and the specification.

The analysis is the same as for the '720 patent. The medium in the '720 patent ***holds*** code. The medium in the '520 patent ***contains*** instructions. Memories and disks are computer-readable media that contain instructions. Transmission media such as coaxial cables, copper wire, and fiber optics, do not "contain" instructions. Considered in its full context, the '520 claim language points to computer-readable medium being "a storage device for use by a computer."

The specification confirms this construction. The '520 patent states:

> Although an exemplary embodiment of the present invention is described as being stored in **memory** 206, one skilled in the art will appreciate that it may also be stored on **other computer-readable media, such as secondary storage devices like hard disks, floppy disks, or CD-Rom**; a carrier wave received from the Internet 204; or other forms of RAM or ROM. Additionally, one skilled in the art will appreciate that computer 202 may contain additional or different components.

'520, 4:48-56. The specification does not disclose coaxial cables, copper wire, fiber optics, or other transmission media as examples of "computer-readable media" or embodiments of the patent, and the claim should not be construed to cover them.

D. '702 patent

Claims 7-12 of the '702 patent are directed to a "computer program product comprising: a **computer usable medium** having computer readable program code embodied therein for pre-processing class files, said computer program product comprising . . . ." '720, 51:43-46. In the context of the '702 claims and specification, this phrase means "a storage device for use by a computer."

Figure 1 of the '702 patent shows the software application (bytecode class files 107) stored on server 100: "Bytecode class files 107 are stored (e.g., in temporary or permanent storage) on server 100, and are available for download over network 101." '702, 3:32-35. The client computer that runs the software application is illustrated in Figure 2. Computer 200 is shown and described as having a main memory 215 that may include DRAM ('702, 6:27-28), mass storage 212 that "may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology" ('702, 6:11-14), and a communications interface 220 that may use a variety of forms of electrical, electromagnetic, or optical signals to transport application code. '702, 6:37-7:14. The client computer may receive application code through the communication interface, and the "received code may be executed by CPU 213 as it is received, and/or stored in mass storage 212, or other non-volatile storage for later execution." '702, 7:10-12.

In the '702 patent, the "computer program product comprising a computer usable medium having computer readable program code embodied therein" is found in the storage devices on the server 100 and the client computer 200. The product is stored on the server, and can be downloaded upon request to the client, where it may be run on by the processor or stored on a local file system. '702, 3:47-59. Although transmission media can be used to transport information and permit downloading, copper wires, coaxial cables, fiber optics, and the like are not part of a "computer program product" and do not have "program code embodied therein." Accordingly, the phrase "computer usable medium" in the '702 patent should not be construed to cover such.

E. '447 and '476 patents

Claims 10-18 of the '447 patent and claims 10-13 and 15-18 of the '476 patent claim a "**computer-readable medium** carrying one or more sequences of one or more instructions . . . ." Claim 14 of the '476 patent claims a "**computer-readable medium** bearing instructions for providing security . . . ."

Like the other patents, the '447 and '476 specifications disclose non-volatile and volatile media such as optical or magnetic disks, storage devices, main memory, floppy disks, hard disks,

magnetic tape, CD-ROMs, RAM, and other physical media. '447, 5:18-25; '476, 5:18-25. ***Unlike*** the other patents, the '447 and '476 patents ***also*** disclose that a "computer-readable medium" may be a transmission medium:

> The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 104 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 110. Volatile media includes dynamic memory, such as main memory 106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 102. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

'447, 5:4-16; '476, 5:4-16. Google will argue that this disclosure means that the computer-readable medium claims are invalid under Section 101 for embracing non-patentable subject matter.

But that is not the right way to analyze this issue. When construing a patent claim term, it is important to give the claim term "the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention, *i.e.*, as of the effective filing date of the patent application." *Phillips*, 415 F.3d at 1313. In December 1997, when the specifications were written, and in 2000 and 2001 when the claims were issued by the PTO after examination, no one thought the claims were not statutory subject matter under Section 101. To the contrary: 1994's *Lowry* and 1995's *Beauregard* ruled the day, and the PTO's considered position was that "a claimed computer-readable medium encoded with a computer program defines structural and functional interrelationships between the computer program and the medium which permit the computer program's functionality to be realized, and is thus statutory." EXAMINATION GUIDELINES FOR COMPUTER-RELATED INVENTIONS 9 (1996) (Peters Decl. Ex. 7).[6] It was not until more than ten years later that the *Nuijten* court held that a "signal per se" did not belong to any of the four enumerated statutory categories: process, machine, manufacture, or composition of matter. Yet the Federal Circuit did not hold that a "computer-readable medium" is not statutory matter, nor would it be expected to.

[6] Available at http://www.uspto.gov/web/offices/pac/dapp/pdf/ciig.pdf.

There is nothing inherent in the phrase "computer-readable medium" that compels a court to conclude that it embraces unpatentable subject matter. There is ample support for the proposition that a computer-readable medium is an article of manufacture or a composition of matter, not only in the intrinsic evidence, but also in the extrinsic evidence. Looking to several contemporary technical dictionaries,[7] Oracle notes that definitions of "medium" and "media" in the computer context encompass storage devices: MICROSOFT PRESS COMPUTER DICTIONARY 301 (3d ed. 1997) (media defined as "The ***physical material***, such as paper, disk, and tape, used for storing computer-based information."); THE ILLUSTRATED DICTIONARY OF ELECTRONICS 430 (7th ed. 1997) (medium defined as "In a computer system, that ***storage*** device onto or into which data is recorded for input into memory (e.g. magnetic disk, magnetic tape, optical disk, etc.)."); THE IEEE STANDARD DICTIONARY OF ELECTRICAL AND ELECTRONICS TERMS 643 (6th ed. 1996) (medium defined as "(1) (computers) The ***material***, or configuration thereof, on which data are recorded; for example, paper tape, cards, magnetic tape."). (Peters Decl. Ex. 8). Each of these definitions falls well within one of the statutory categories and describes a medium having a non-transitory character: a physical material, a storage device, or a material.

Given the choice to construe an issued patent claim to cover statutory or nonstatutory subject matter, it would be unreasonable choose to construe the claim to cover nonstatutory subject matter and risk invalidating the claim. Such a construction would not comport with the intent of the inventor or the PTO's issuance of the patent. The principle that "claims should be so construed, if possible, as to sustain their validity" applies here. *Rhine v. Casio, Inc.*, 183 F.3d 1342, 1345 (Fed Cir. 1999); *see also Astra Aktiebolag v. Andrx Pharms., Inc.*, 222 F. Supp. 2d 423, 458 (S.D.N.Y. 2002) ("Whenever a claim is susceptible to one construction that would render it valid and another construction that would render it invalid, the claim will be construed to sustain its validity."). The disclosure of transmission media in the '447 and '476 specifications does not bar application of the principle. Even when the specification contains an express

[7] "Dictionaries or comparable sources are often useful to assist in understanding the commonly understood meaning of words and have been used both by our court and the Supreme Court in claim interpretation." *Phillips*, 415 F.3d at 1322.

definition of a claim term, a court may construe it more narrowly. *See, e.g., Trading Techs. Int'l, Inc. v. eSpeed, Inc.*, 595 F.3d 1340, 1353-55 (Fed. Cir. 2010) (holding that, despite an express definition in the specification, the district court was correct to make "two important changes to this express definition in construing the word"); *Ecolab, Inc. v. FMC Corp.*, 569 F.3d 1335, 1344-45 (Fed. Cir. 2009) (holding that, although the patent specification provided an express definition of "sanitize" (it "denote[s] a bacterial population reduction to a level that is safe for human handling and consumption"), the term should be construed to "mean that the treated meat has become safe for human handling and post-cooking consumption.").

The Court should construe "computer-readable medium" in a way that sustains its validity. As the Supreme Court held in *Warner-Jenkinson* and reaffirmed in *Festo*: "To change so substantially the rules of the game now could very well subvert the various balances the PTO sought to strike when issuing the numerous patents which have not yet expired and which would be affected by our decision." *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 739 (2002) (quoting *Warner-Jenkinson Co. v. Hilton Davis Chem. Co.*, 520 U.S. 17, 32 n.6 (1997)). As the Court considers how to construe PTO-examined and -issued Beauregard claims, each entitled to a presumption of validity, it should do so in a manner that does not "risk destroying the legitimate expectations of inventors in their property" or "unfairly discount the expectations of a patentee who had no notice at the time of patent prosecution." *Id.*

Dated: March 17, 2011

MICHAEL A. JACOBS
MARC DAVID PETERS
MORRISON & FOERSTER LLP

By: /s/ Michael A. Jacobs

Attorneys for Plaintiff
ORACLE AMERICA, INC.